Methods

• consists of method name, parameters, return value type, and body

```
public static void main(String[] args){
    // body
}
```

• signature of method = method name and parameters list

can have multiple methods with same name as long as different signature

call method overloading

```
public static int findMax(int num1, int num2){
    int maxNum = (num1 > num2) ? num1 : num2
    return maxNum;
}
```

```
public static double findMax(double num1, double num2){
    double maxNum = (num1 > num2) ? num1 : num2
    return maxNum;  // maxNum now a double
}
```

• if method has no return value, type is void
• a method can have no parameters  `myString.length()`
• public means method can be called from other classes  e.g. `Math.sqrt(x)`
Method Flow of Control

```java
public class Demo{
    public static void main(String[] args){
        // set up arguments
        int a=4, b=8;
        int getMax = findMax(a, b);
        System.out.println(…..);
    }

    public static int findMax(int num1, int num2){
        int maxNum = (num1 > num2) ? num1 : num2
        return maxNum;
    }
}
```

- System keeps track of variables and parameters in an activation record / frame.
- When a method is called, a new activation record is created, and the old method’s record remains.
- Activation records are a stack.

Can also use expressions as arguments, e.g., `findMax(2*a+3, draw stack, what if variables have same name?)`
Method Arguments

```java
public static void main(String[] args){
    int a=4, b=8;
    System.out.println("from main:" + a + b);
    swap(a,b);
    System.out.println("from main:" + a + b);
}

public static void swap(int num1, int num2){
    System.out.println("from swap:" + num1 + num2);
    int tmp = num1;
    num1 = num2;
    num2 = tmp;
    System.out.println("from swap:" + num1 + num2);
    return;
}
```

• what is output of program?
• arguments are passed by value
• to get a value back out from method use return value
Scope of Variables

```java
public static void swap(int num1, int num2){
    System.out.println("from swap:" + num1 + num2);
    int tmp = num1;
    num1 = num2;
    num2 = tmp;
    System.out.println("from swap:" + num1 + num2);
    return;
}
```

- **scope of variable** is part of program where it can be referenced
- **scope** is from its declaration to end of block that contains it.
- different methods can have same variable name if they are in different blocks (not nested blocks or nested methods - we haven’t studied this yet)
- in for loop above, \( i \) keep its value through loop, \( j \) is new each time.
- scope of \( i \) is entire loop, scope of \( j \) ends at end of block, \( sum \) can be referenced outside block
Scope of Variables

```java
class Scope {
    public static void main (String[] args) {
        int x = 0;
        int y = 0;
        for (int i = 1; i < 10; i++) {
            int x = 0;
            x += i;
        }
    }
}
```

is this code OK?